

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

**Кафедра системного програмування та спеціалізованих комп’ютерних систем**

**Лабораторна робота №2**

з дисципліни **Бази даних і засоби управління**

*на тему: “Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL”*

Виконала:

студентка ІII курсу

групи КВ-02

Рагозіна К. О.

Перевірив:

Павловський В. І.

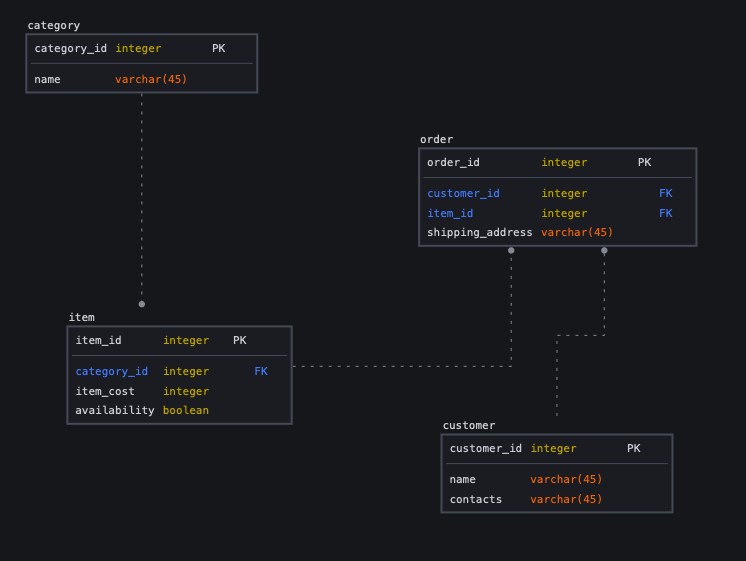
Київ – 2023

*Метою роботи* є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

*Загальне завдання роботи* полягає у наступному:

1. Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

**Логічна модель предметної області «Онлайн магазин»**

Рисунок 2 - Схема бази даних (інструмент: sqldbm.com)

**Середовище та компоненти розробки**

Для виконання лабораторної роботи використовувалась мова програмування C# та середовище розробки Visual Studio Code. Для підключення до серверу бази даних PostgreSQL використано сторонню бібліотеку Npgsql.

**Шаблон проектування**

MVC – шаблон проектування, який використаний у програмі.

Model – представляє клас, що описує логіку використовуваних даних. Згідно компоненту моделі, у моїй програмі відповідають всі компоненти які знаходять у папці Models.

View – консольний інтерфейс, з яким буде взаємодіяти користувач. Містить компоненти, згідно яким користувач бачить необхідні дані, що є представленням даних у вигляді консольного інтерфейсу.

Controller – представляє клас, що забезпечує зв'язок між користувачем і системою, поданням і сховищем даних. Отримує введені користувачем дані і обробляє їх, в залежності від результатів обробки відправляє користувачеві певний висновок.

**Структура програми та опис**

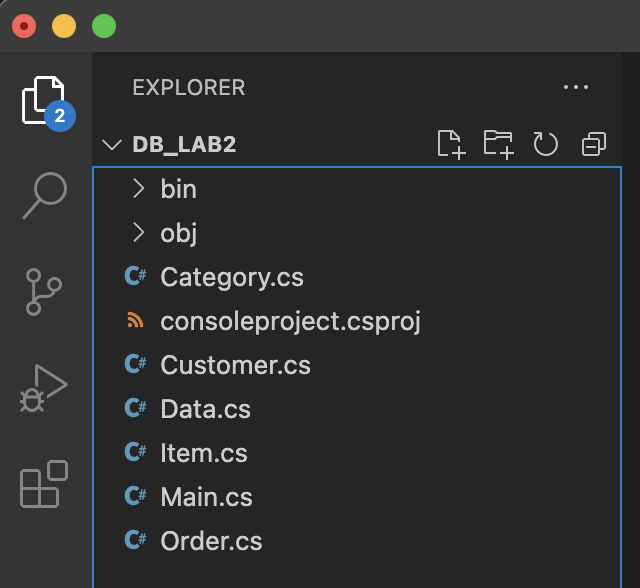
Структура програми має вигляд:

Рисунок 2 – структура програми

Програма має 6 модулів:

1. Main.cs – модуль, що викликає методи з контролерів та містить засоби для обробки виключень та повідомлення про помилки;
2. Data.cs – модуль, що містить функції та засоби для підключення бази даних до програми;
3. Item.cs – контролер для таблиці Item;
4. Category.cs – контролер для таблиці Category;
5. Order.cs – контролер для таблиці Order;
6. Customer.cs – контролер для таблиці Customer;

**Меню програми**

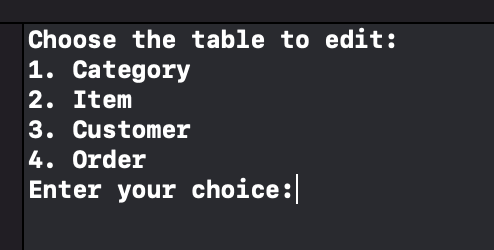
Головне меню програми, у якому користувач обирає компонент БД, із яким надалі буде працювати, має вигляд:

Рисунок 3 – Головне меню програми

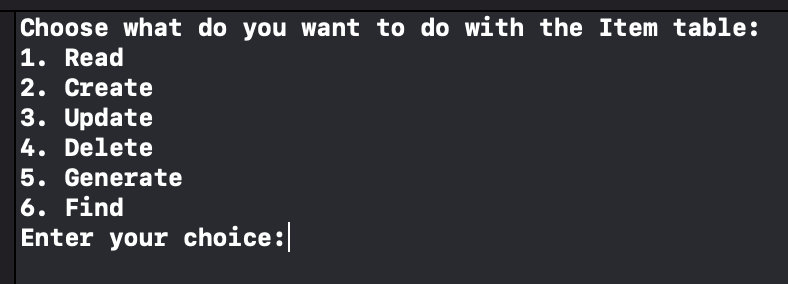
Меню, у якому користувач обирає дії, яку він хоче зробити із обраною категорією має вигляд:

Рисунок 4 – Меню обирання бажаної дії

Розглянемо функціональні можливості програми:

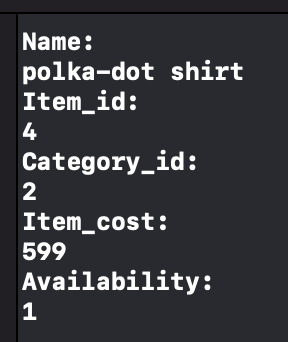
1. Додавання даних до БД

Рисунок 5 – Додавання даних про товар до БД

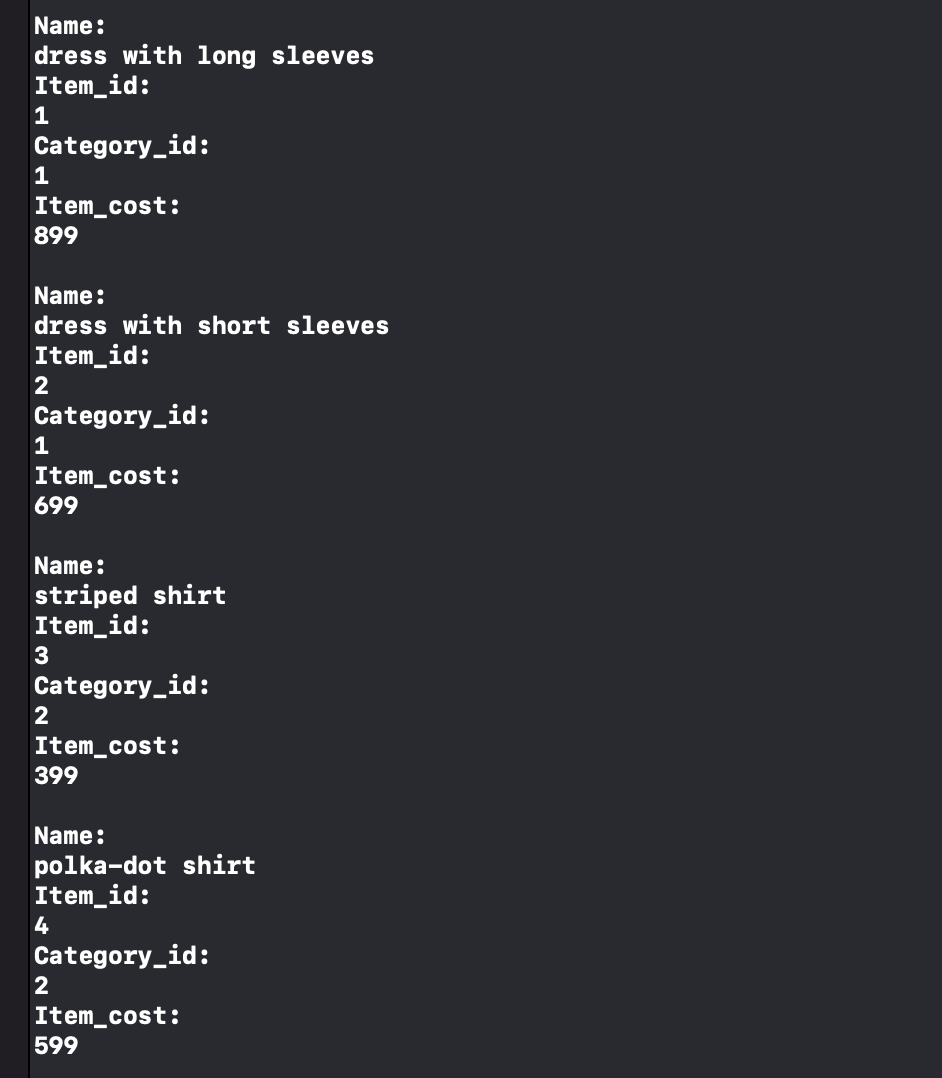


Рисунок 6 – Результат операції додавання товарів

SQL-запит:

string sqlInsert = "Insert into Item(Name, Item\_Id, Category\_Id, Item\_cost) VALUES(@Name, @Item\_Id, @Category\_Id, @Item\_cost)";

Для контролю наявності відповідного рядка у батьківській таблиці під час внесення даних до дочірньої таблиці в коді передбачена спеціальна процедура:

try { cmd.ExecuteNonQuery (); }

catch ( EXCEPTION ex )

{

Console.WriteLine (ex.Message);

Console.ReadLine ();

}

finally

{

sqlConnection.Close ();

}

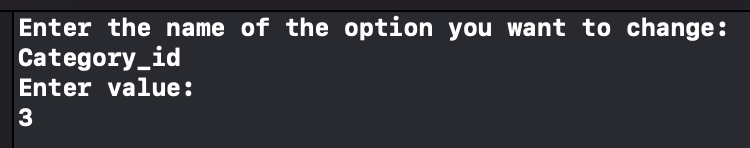
1. Редагування даних

Рисунок 7 – Зміна параметру Category\_id

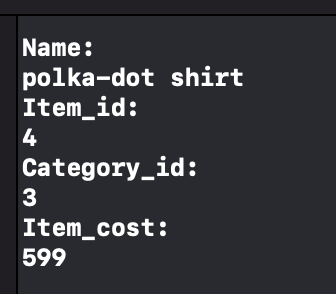


Рисунок 8 – Результат операції зміни даних про параметр Category\_id

SQL-запит:

string sqlUpdate = "Update @table set @field\_to\_update = @new\_value where @field\_to\_find = @old\_value";

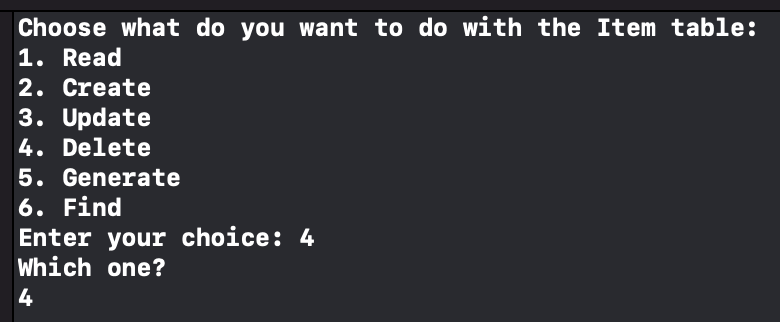
1. Видалення даних

Рисунок 9 – Видалення товару з Item\_id – 4 з категорії Item

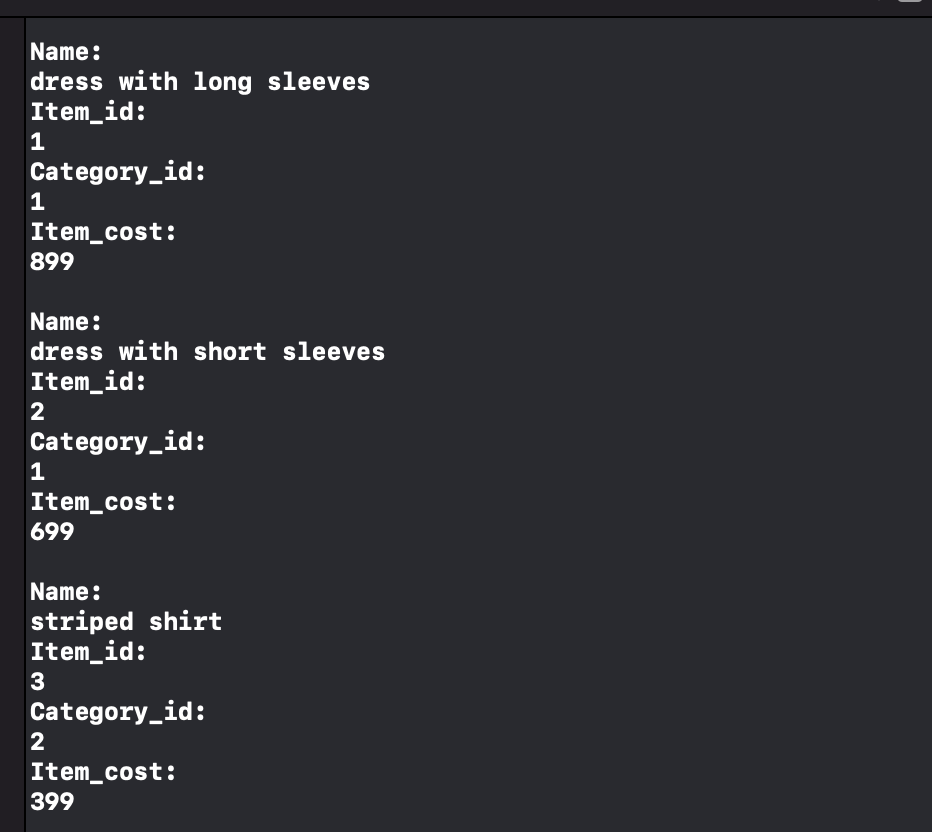


Рисунок 10 - Товар з Item\_id – 4 видалено з категорії Item

SQL-запит:

base.Delete ("Delete from Item where Item\_Id = ");

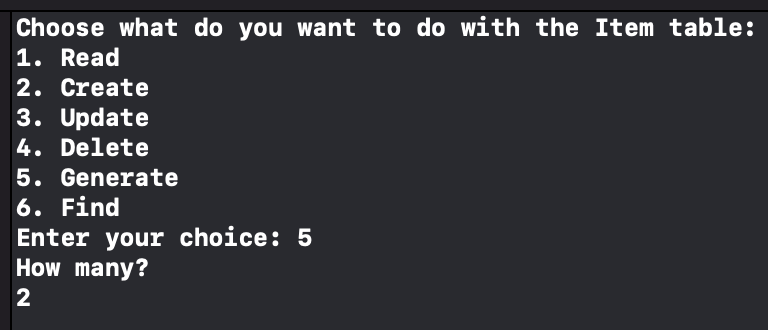
1. Генерація товарів

Рисунок 11 – Генерування двох товарів

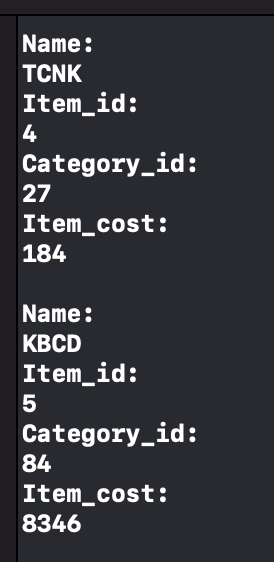


Рисунок 12 – Результат генерування товарів

SQL-запит:

string sqlGenerate = "Insert into Item(Name, Item\_Id, Category\_Id, Item\_cost) (select " + base.sqlRandomString + ", " + base.sqlRandomInteger + ", " + base.sqlRandomInteger + ", " + base.sqlRandomInteger + ", " base.Generate (sqlGenerate)

; PUBLIC readonly string sqlUpdate = "Update @table set @field\_to\_update = @new\_value where @field\_to\_find = @old\_value"

; PUBLIC readonly string sqlRandomString = "chr(trunc(65 + random() \* 50)::int) || chr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int)"

; PUBLIC readonly string sqlRandomInteger = "trunc(random()\*1000)::int"

; PUBLIC readonly string sqlRandomDate = "timestamp '00:10:59' + random() \* (timestamp '00:10:59' - timestamp '00:05:00')"

;

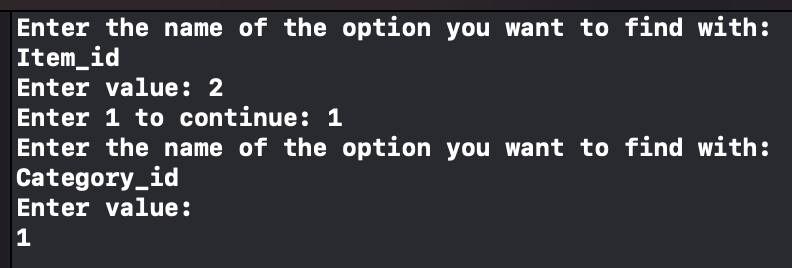
1. Пошук за декількома атрибутами одночасно

Рисунок 13 – Пошук товару за декількома атрибутами



Рисунок 14 – Результат пошуку товару за декількома атрибутами

**Лістинг програми**

**Main.cs**

using bd\_lab2.Controllers;

using System;

namespace bd\_lab2{

    class Program{

        static void Main(string[] args){

            string connectionString = "Host=localhost; Username=postgres; Password=4545; Database=BD";

            int table = 0;

            int action = 0;

            do{

                table = FirstMenu();

                if (table == 0){

                    return;

                }

                BaseController controller = null;

                switch (table){

                    case 1:

                        action = SecondMenu("Category");

                        controller = new Category(connectionString);

                        break;

                    case 2:

                        action = SecondMenu("Item");

                        controller = new Item(connectionString);

                        break;

                    case 3:

                        action = SecondMenu("Customer");

                        controller = new Customer(connectionString);

                        break;

                    case 4:

                        action = SecondMenu("Order");

                        controller = new Recording\_studio(connectionString);

                        break;

                }

                switch (action){

                    case 1:

                        controller.Read();

                        break;

                    case 2:

                        controller.Create();

                        break;

                    case 3:

                        controller.Update();

                        break;

                    case 4:

                        controller.Delete();

                        break;

                    case 5:

                        controller.Generate();

                        break;

                    case 6:

                        controller.Find();

                        break;

                }

            } while (true);

        }

        public static int FirstMenu(){

            var choice = 0;

            var correct = false;

            do{

                Console.Clear();

                Console.WriteLine("Choose the table to edit:");

                Console.WriteLine("1. Category");

                Console.WriteLine("2. Item");

                Console.WriteLine("3. Customer");

                Console.WriteLine("4. Order");

                correct = Int32.TryParse(Console.ReadLine(), out choice);

            } while (choice < 0 || choice > 4 || correct == false);

            return choice;

        }

        public static int SecondMenu(string tableToChange){

            var choice = 0;

            var correct = false;

            do{

                Console.Clear();

                Console.WriteLine("Choose what do you want to do with the'" + tableToChange + "' table:");

                Console.WriteLine("1.Read");

                Console.WriteLine("2.Create");

                Console.WriteLine("3.Update");

                Console.WriteLine("4.Delete");

                Console.WriteLine("5.Generate");

                Console.WriteLine("6.Find");

                correct = Int32.TryParse(Console.ReadLine(), out choice);

            } while (choice < 0 || choice > 6 || correct == false);

            return choice;

        }

    }

}

**Data.cs**

using Npgsql;

using System;

using System.Collections.Generic;

using System.Text;

namespace bd\_lab2.Controllers{

    public abstract class Data{

        public string connectionString;

        protected NpgsqlConnection sqlConnection;

        string fieldToFind = null;

        string valueToFind = null;

        string fieldToSet = null;

        string valueToSet = null;

        string[] fieldsToFind = new string[10];

        string[] valuesToFind = new string[10];

        public readonly string sqlUpdate = "Update @table set @field\_to\_update = @new\_value where @field\_to\_find = @old\_value";

        public readonly string sqlRandomString = "chr(trunc(65 + random() \* 50)::int) || chr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int)";

        public readonly string sqlRandomInteger = "trunc(random()\*1000)::int";

        public Data(string connectionString){

            this.connectionString = connectionString;

            this.sqlConnection = new NpgsqlConnection(connectionString);

        }

        public void Read(){

            Read("");

        }

        public virtual void Create(){

            throw new NotImplementedException();

        }

        public virtual void Update(){

            throw new NotImplementedException();

        }

        public virtual void Delete(){

            throw new NotImplementedException();

        }

        virtual public void Generate(){

            throw new NotImplementedException();

        }

        public virtual void Find(){

            Console.Clear();

            int actualSize = 0;

            for (int i = 0; i < 10; i++){

                Console.WriteLine("Enter the name of the option you want to find with:");

                fieldsToFind[i] = Console.ReadLine();

                Console.WriteLine("Enter value:");

                valuesToFind[i] = Console.ReadLine();

                Console.WriteLine("Enter 1 to continue:");

                actualSize++;

                int choose = 0;

                bool correct = Int32.TryParse(Console.ReadLine(), out choose);

                if (correct = false || choose != 1){

                    break;

                }

            }

            string whereCondition = " where ";

            int parseInt;

            if (Int32.TryParse(valuesToFind[0], out parseInt) == false){

                valuesToFind[0] = "'" + valuesToFind[0] + "'";

            }

            whereCondition +=  fieldsToFind[0] + " = " + valuesToFind[0];

            for (int i = 1; i < actualSize; i++){

                if (Int32.TryParse(valuesToFind[i], out parseInt) == false){

                    valuesToFind[i] = "'" + valuesToFind[i] + "'";

                }

                whereCondition += " and " + fieldsToFind[i] + " = " + valuesToFind[i];

            }

            Read(whereCondition);

        }

        virtual public void Read(string whereCondition){

        }

        protected void Delete(string sqlDelete){

            bool correct = false;

            int id = 0;

            do{

                Console.WriteLine("Which one?");

                correct = Int32.TryParse(Console.ReadLine(), out id);

                if (correct == false){

                    Console.WriteLine("Error, try again");

                    Console.ReadLine();

                    continue;

                }

            } while (correct == false || id < 0);

            sqlConnection.Open();

            using var cmd = new NpgsqlCommand(sqlDelete + id, sqlConnection);

            try{

                cmd.Prepare();

                cmd.ExecuteNonQuery();

            }

            catch (Exception ex){

                Console.WriteLine(ex.Message);

                Console.ReadLine();

            }

            finally{

                sqlConnection.Close();

            }

        }

        private void Update(string table, string field\_to\_update, string new\_value, string field\_to\_find, string old\_value){

            sqlConnection.Open();

            StringBuilder updateString = new StringBuilder("Update", 200);

            int new\_int;

            if (!Int32.TryParse(new\_value, out new\_int)){

                new\_value = "'" + new\_value + "'";

            }

            if (!Int32.TryParse(old\_value, out new\_int)){

                old\_value = "'" + old\_value + "'";

            }

            updateString.AppendFormat(" {0} set {1} = {2} where {3} = {4}", table, field\_to\_update, new\_value, field\_to\_find, old\_value);

            using var cmd = new NpgsqlCommand(updateString.ToString(), sqlConnection);

            try{

                cmd.Prepare();

                cmd.ExecuteNonQuery();

            }

            catch (Exception ex){

                Console.WriteLine(ex.Message);

                Console.ReadLine();

            }

            finally{

                sqlConnection.Close();

            }

        }

        protected void Update(string sqlUpdate){

            Console.Clear();

            Console.WriteLine("Enter the name of the option you want to find:");

            fieldToFind = Console.ReadLine();

            Console.WriteLine("Enter value:");

            valueToFind = Console.ReadLine();

            Console.WriteLine("Enter the name of the option you want to change:");

            fieldToSet = Console.ReadLine();

            Console.WriteLine("Enter value:");

            valueToSet = Console.ReadLine();

            int ParseInt = 0;

            if (Int32.TryParse(valueToFind, out ParseInt) == false){

                valueToFind = "'" + valueToFind + "'";

            }

            if (Int32.TryParse(valueToSet, out ParseInt) == false){

                valueToSet = "'" + valueToSet + "'";

            }

            string sqlQuery = sqlUpdate + "set " + fieldToSet + " = " + valueToSet + " where " + fieldToFind + " = " + valueToFind;

            sqlConnection.Open();

            using var cmd = new NpgsqlCommand(sqlQuery, sqlConnection);

            try{

                cmd.Prepare();

                cmd.ExecuteNonQuery();

            }

            catch (Exception ex){

                Console.WriteLine(ex.Message);

                Console.ReadLine();

            }

            finally{

                sqlConnection.Close();

            }

        }

        protected void Generate(string sqlGenerate){

            sqlConnection.Open();

            using var cmd = new NpgsqlCommand(sqlGenerate, sqlConnection);

            try{

                cmd.Prepare();

                cmd.ExecuteNonQuery();

            }

            catch (Exception ex){

                Console.WriteLine(ex.Message);

                Console.ReadLine();

            }

            finally{

                sqlConnection.Close();

            }

        }

    }

}

**Category.cs**

using Npgsql;

using System;

using System.Collections.Generic;

using System.Text;

namespace bd\_lab2.Controllers{

    public class Category : Data{

        public Category(string connectionString) : base(connectionString) { }

        public override void Read(string whereCondition){

            Console.Clear();

            sqlConnection.Open();

            string sqlSelect = "select Category\_Id, Name, CreationDate from Category";

            using var cmd = new NpgsqlCommand(sqlSelect + whereCondition, sqlConnection);

            try{

                using NpgsqlDataReader rdr = cmd.ExecuteReader();

                while (rdr.Read()){

                    Console.WriteLine("Category\_Id: {0}", rdr.GetValue(0));

                    Console.WriteLine("Name: {0}", rdr.GetValue(1));

                    Console.WriteLine("CreationDate: {0}", rdr.GetValue(3));

                    Console.WriteLine();

                }

                Console.WriteLine();

            }

            catch (Exception ex){

                Console.WriteLine(ex.Message);

                Console.ReadLine();

            }

            finally{

                sqlConnection.Close();

            }

            Console.ReadLine();

        }

        public override void Create(){

            string sqlInsert = "Insert into Category(Name, CreationDate) VALUES(@Name, @CreationDate)";

            string Name = null;

            DateTime CreationDate = 0;

            bool correct = false;

            do{

                Console.Clear();

                Console.WriteLine("Name:");

                Name = Console.ReadLine();

                if (name.Length > 20){

                    correct = false;

                    Console.WriteLine("Error, try again");

                    Console.ReadLine();

                    continue;

                }

                Console.WriteLine("CreationDate:");

                correct = DateTime.TryParse(Console.ReadLine(), out CreationDate);

                if(correct == false){

                    Console.WriteLine("Error, try again");

                    Console.ReadLine();

                }

                correct = true;

            } while (correct == false);

            sqlConnection.Open();

            using var cmd = new NpgsqlCommand(sqlInsert, sqlConnection);

            cmd.Parameters.AddWithValue("Name", Name);

            cmd.Parameters.AddWithValue("CreationDate", CreationDate);

            cmd.Prepare();

            try{

                cmd.ExecuteNonQuery();

            }

            catch (Exception ex){

                Console.WriteLine(ex.Message);

                Console.ReadLine();

            }

            finally{

                sqlConnection.Close();

            }

        }

        public override void Delete(){

            base.Delete("Delete from Category where Category\_Id = ");

        }

        public override void Update(){

            base.Update("Update Band ");

        }

        public override void Generate(){

            Console.WriteLine("How many?");

            bool correct = false;

            int recordsAmount;

            correct = Int32.TryParse(Console.ReadLine(), out recordsAmount);

            string sqlGenerate = "Insert into categories(Name, CreationDate) (select "

                + base.sqlRandomString

                + ", "

                + base.sqlRandomString

                + ", ";

            base.Generate(sqlGenerate);

        }

    }

}

**Item.cs**

using Npgsql;

using System;

using System.Collections.Generic;

using System.Text;

namespace bd\_lab2.Controllers{

    public class Item : Data{

        public Item(string connectionString) : base(connectionString) { }

        public override void Read(string whereCondition){

            Console.Clear();

            sqlConnection.Open();

            string sqlSelect = "select Name, Item\_Id, Category\_Id, Item\_cost,";

            using var cmd = new NpgsqlCommand(sqlSelect + whereCondition, sqlConnection);

            try{

                using NpgsqlDataReader rdr = cmd.ExecuteReader();

                while (rdr.Read()){

                    Console.WriteLine("Name: {0}", rdr.GetValue(0));

                    Console.WriteLine("Item\_Id: {0}", rdr.GetValue(1));

                    Console.WriteLine("Category Id: {0}", rdr.GetValue(2));

                    Console.WriteLine("Item\_cost: {0}", rdr.GetValue(3));

                    Console.WriteLine();

                }

                Console.WriteLine();

            }

            catch (Exception ex){

                Console.WriteLine(ex.Message);

                Console.ReadLine();

            }

            finally{

                sqlConnection.Close();

            }

            Console.ReadLine();

        }

        public override void Create(){

            string sqlInsert = "Insert into Item(Name, Item\_Id, Category\_Id, Item\_cost) VALUES(@Name, @Item\_Id, @Category\_Id, @Item\_cost)";

            string Name = null;

            int Category\_id = 0;

            int Item\_cost = int;

            bool correct = false;

            do{

                Console.Clear();

                Console.WriteLine("Name:");

                name = Console.ReadLine();

                if (name.Length > 30){

                    correct = false;

                    Console.WriteLine("Error, try again");

                    Console.ReadLine();

                    continue;

                }

                Console.WriteLine("Item\_id:");

                correct = Int32.TryParse(Console.ReadLine(), out Item\_Id);

                if (correct == false){

                    Console.WriteLine("Error, try again");

                    Console.ReadLine();

                }

                Console.WriteLine("Category\_id:");

                correct = Int32.TryParse(Console.ReadLine(), out Category\_Id);

                if (correct == false){

                    Console.WriteLine("Error, try again");

                    Console.ReadLine();

                }

                Console.WriteLine("Item\_cost:");

                correct = Int32.TryParse(Console.ReadLine(), out Item\_cost);

                if (correct == false){

                    Console.WriteLine("Error, try again");

                    Console.ReadLine();

                }

                correct = true;

            } while (correct == false);

            sqlConnection.Open();

            using var cmd = new NpgsqlCommand(sqlInsert, sqlConnection);

            cmd.Parameters.AddWithValue("Name", Name);

            cmd.Parameters.AddWithValue("Item\_Id", Item\_Id);

            cmd.Parameters.AddWithValue("Category\_Id", Category\_Id);

            cmd.Parameters.AddWithValue("Item\_cost", Item\_cost);

            cmd.Prepare();

            try{

                cmd.ExecuteNonQuery();

            }

            catch (Exception ex){

                Console.WriteLine($"Cannot be created: {ex.Message}");

                Console.ReadLine();

            }

            finally{

                sqlConnection.Close();

            }

        }

        public override void Delete(){

            base.Delete("Delete from Item where Item\_Id = ");

        }

        public override void Update(){

            base.Update("Update Item: ");

        }

        public override void Generate(){

            Console.WriteLine("How many?");

            bool correct = false;

            int recordsAmount;

            correct = Int32.TryParse(Console.ReadLine(), out recordsAmount);

            string sqlGenerate = "Insert into Item(Name, Item\_Id, Category\_Id, Item\_cost) (select "

            + base.sqlRandomString

            + ", "

            + base.sqlRandomInteger

            + ", " + base.sqlRandomInteger

            + ", " + base.sqlRandomInteger

            + ", " base.Generate (sqlGenerate);

        }

    }

}

**Customer.cs**

using Npgsql;

using System;

using System.Collections.Generic;

using System.Text;

namespace bd\_lab2.Controllers{

    public class Customer : Data{

        public Customer(string connectionString) : base(connectionString) { }

        public override void Read(string whereCondition){

            Console.Clear();

            sqlConnection.Open();

            string sqlSelect = "select Customer\_Id, Name, Contacts";

            using var cmd = new NpgsqlCommand(sqlSelect + whereCondition, sqlConnection);

            try{

                using NpgsqlDataReader rdr = cmd.ExecuteReader();

                while (rdr.Read()){

                    Console.WriteLine("Customer\_Id: {0}", rdr.GetValue(0));

                    Console.WriteLine("Name: {0}", rdr.GetValue(1));

                    Console.WriteLine("Contacts: {0}", rdr.GetValue(2));

                    Console.WriteLine();

                }

                Console.WriteLine();

            }

            catch (Exception ex){

                Console.WriteLine(ex.Message);

                Console.ReadLine();

            }

            finally{

                sqlConnection.Close();

            }

            Console.ReadLine();

        }

        public override void Create(){

            string sqlInsert = "Insert into Customer(Customer\_Id, Name, Contacts) VALUES(@Customer\_Id, @Name, @Contacts)";

            string Customer\_Id = 0;

            int Name = null;

            int Contacts = null;

            bool correct = false;

            do{

                Console.Clear();

                Console.WriteLine("Name:");

                name = Console.ReadLine();

                if (name.Length > 40){

                    correct = false;

                    Console.WriteLine("Error, try again");

                    Console.ReadLine();

                    continue;

                }

                Console.WriteLine("Contacts:");

                correct = Int32.TryParse(Console.ReadLine());

                if (name.Length > 40){

                    Console.WriteLine("Error, try again");

                    Console.ReadLine();

                }

                correct = true;

            } while (correct == false);

            sqlConnection.Open();

            using var cmd = new NpgsqlCommand(sqlInsert, sqlConnection);

            cmd.Parameters.AddWithValue("Name", Name);

            cmd.Parameters.AddWithValue("Contacts", Contacts);

            cmd.Prepare();

            try{

                cmd.ExecuteNonQuery();

            }

            catch (Exception ex){

                Console.WriteLine(ex.Message);

                Console.ReadLine();

            }

            finally{

                sqlConnection.Close();

            }

        }

        public override void Delete(){

            base.Delete("Delete from Customer where Customer\_Id = ");

        }

        public override void Update(){

            base.Update("Update Customer ");

        }

        public override void Find(){

            base.Find();

        }

        public override void Generate(){

            Console.WriteLine("How many?");

            bool correct = false;

            int recordsAmount;

            correct = Int32.TryParse(Console.ReadLine(), out recordsAmount);

            string sqlGenerate = "Insert into Customer(Name, Contacts) (select "

                + base.sqlRandomString

                + ", "

                + base.sqlRandomString

                + ", ";

            base.Generate(sqlGenerate);

        }

    }

}

**Order.cs**

using Npgsql;

using System;

using System.Collections.Generic;

using System.Text;

namespace bd\_lab2.Controllers{

    public class Order : Data{

        public Order(string connectionString) : base(connectionString) { }

        public override void Read(string whereCondition){

            Console.Clear();

            sqlConnection.Open();

            string sqlSelect = "select Order\_Id, Shipping\_address from Order";

            using var cmd = new NpgsqlCommand(sqlSelect + whereCondition, sqlConnection);

            try{

                using NpgsqlDataReader rdr = cmd.ExecuteReader();

                while (rdr.Read()){

                    Console.WriteLine("Order\_Id: {0}", rdr.GetValue(0));

                    Console.WriteLine("Shipping address: {0}", rdr.GetValue(1));

                    Console.WriteLine();

                }

                Console.WriteLine();

            }

            catch (Exception ex){

                Console.WriteLine(ex.Message);

                Console.ReadLine();

            }

            finally{

                sqlConnection.Close();

            }

            Console.ReadLine();

        }

        public override void Create(){

            string sqlInsert = "Insert into Order (Order\_id, Shipping\_address) VALUES(@Order\_Id, @Shipping\_address)";

            int Order\_Id= 0;

            string Shipping\_address = null;

            bool correct = false;

            do{

                Console.Clear();

                Console.WriteLine("Order\_Id:");

                correct = Int32.TryParse(Console.ReadLine(), out Order\_Id);

                if (correct == false){

                    Console.WriteLine("Error, try again");

                    Console.ReadLine();

                }

                Console.WriteLine("Shipping address:");

                name = Console.ReadLine();

                if (name.Length > 50){

                    correct = false;

                    Console.WriteLine("Error, try again");

                    Console.ReadLine();

                    continue;

                }

                correct = true;

            } while (correct == false);

            sqlConnection.Open();

            using var cmd = new NpgsqlCommand(sqlInsert, sqlConnection);

            cmd.Parameters.AddWithValue("Customer\_Id", customer\_id);

            cmd.Parameters.AddWithValue("Shipping\_address", Name);

            cmd.Prepare();

            try{

                cmd.ExecuteNonQuery();

            }

            catch (Exception ex){

                Console.WriteLine(ex.Message);

                Console.ReadLine();

            }

            finally{

                sqlConnection.Close();

            }

        }

        public override void Delete(){

            base.Delete("Delete from Order where Rec\_Id = ");

        }

        public override void Update(){

            base.Update("Update Order ");

        }

        public override void Generate(){

            Console.WriteLine("How many?");

            bool correct = false;

            int recordsAmount;

            correct = Int32.TryParse(Console.ReadLine(), out recordsAmount);

            string sqlGenerate = "Insert into Order(Shipping\_address) (select "

            + base.sqlRandomString

            + ", " ;

            base.Generate(sqlGenerate);

        }

    }

}

**Посилання на GitHub**

[**https://github.com/kristinakunica/DB\_lab2**](https://github.com/kristinakunica/DB_lab2)